

컴퓨터는 우리의 명령을 어떻게 알아들을까?

2011. 11. 2 정동욱

1. 계산기와 컴퓨터의 차이

컴퓨터는 다양한 프로그램을 바꾸어 수행함으로써 정해진 계산 이상의 업무를 처리할 수 있다.

2. 프로그램이란? 정보를 처리하기 위한 순서를 컴퓨터에게 알려주는 일종의 '지침서'

프로그램 없이는 컴퓨터는 아무 일도 하지 않음 (프로그램과 자료를 통틀어 소프트웨어라고 함)

3. 컴퓨터는 '기계어'밖에 모른다

컴퓨터에게 일을 시키려면 컴퓨터가 알아들을 수 있는 언어, 즉 프로그래밍 언어로 지시를 주어야 함. 사실 컴퓨터가 알아들을 수 있는 언어는 0과 1로 조합된 기계어밖에 없음. 예를 들어 덧셈 명령은 그에 해당하는 0110과 같은 코드로 되어 있음. (알테어 컴퓨터의 경우 스위치를 이용해 이러한 코드를 입력할 수 있었음. 도중에 하나라도 틀리면 어떻게? 처음부터 다시 해야 T.T.)

3-1. 어셈블리어

0과 1의 기계어 명령어를 짧은 영어 단어로 바꾸어 좀더 프로그램을 만들기 쉽도록 한 언어. 예를 들어 AD = 0110, LD(불러오기), ST(저장하기), HALT(종료하기). [이런 명령어 집합은 어셈블리마다 다르고, CPU마다 다름]

4. 고급 언어

어셈블리어란 기계어를 알파벳 기호로 바꾼 것에 불과하기 때문에, 예를 들어 '2+3=?'과 같은 계산조차도 기계에 일을 시키려면 생각보다 훨씬 복잡하게 일을 시켜야 함(CPU 작동 이해 부분에서 다시).

이를 개선하여 '2+3의 계산을 하라'와 같은 보다 인간과 가까운 명령들로 프로그램을 쓸 수 있도록 만든 것이 고급 언어(예: BASIC, FORTRAN, COBOL, C, Java, php 등). 고급언어로 만든 프로그램은 '인터프리터'나 '컴파일러'라고 하는 또다른 프로그램에 의해 기계어로 번역되어야 진짜 컴퓨터가 일을 수행할 수 있는 프로그램으로 바뀜(둘을 구분하여, 전자를 소스코드, 후자를 프로그램이라고 하기도 함).

※ 인터프리터와 컴파일러의 차이는? 전자는 한줄 한줄 동시통역, 후자는 통번역. 장단점은? 인터프리터의 경우 프로그램의 문제를 쉽게 알아볼 수 있음. 그러나 속도 느리고 프로그램의 크기도 큼. 요즘엔 대부분 컴파일러 사용. (과거의 PC에는 베이직이 탑재되어 있어, 모든 컴퓨터 사용자가 직접 프로그래밍을 했음. 적어도 프로그래밍을 직접 하지 않더라도 남의 소스코드를 받아서 보긴 해야 했음.)

5. 왜 프로그래밍은 어려운가

컴퓨터는 모호한 명령을 알아듣지 못함. 순차적으로 명료하게 명령을 내려주어야만 일을 제대로 수행. 컴퓨터 로봇은 '거기 있는 컵 좀 이리로 갖다 줘'라는 명령을 알아들을 수 없음. 훨씬 긴 목록의 세세한 명령을 내려야 할 것임.

6. CPU의 작동과 어셈블리어 프로그램의 작동

간단한 예로, 1+2를 어셈블리어로 명령을 시켜보고, 그 수행을 살펴보자(LD 10, AD 11, ST 12, HALT).

이 작업에 무려 20번의 작업이! 물론 작업 하나당 무지 빠르지만(초당 몇억~몇조번), 프로그램에 커지면 문제가 됨.

7. 알고리즘

알고리즘과 프로그램의 차이 : 프로그램이 명령 '지시서'라면, 알고리즘은 그 '지시서'가 구현하고 있는 추상적인 해법의 아이디어. 스토리와 책의 차이와 유사. 소설책이 다른 언어로 번역되더라도 스토리는 변하지 않듯이, 알고리즘도 마찬가지. 프로그램이 다른 언어로 번역되더라도, 그것이 구현하고 있는 알고리즘은 변하지 않음.

알고리즘은 추상적 아이디어이기 때문에, 여러 방식으로 표현될 수 있음. 실제 프로그램 코드로 표현될 수도 있지만, 흔히 세부적인 사항을 버린 채 아이디어를 명확하게 전달할 수 있는 유사코드(pseudocode) 형태로 작성됨. 아래는 많이 사용되는 표현들

배정문 : 이름 \leftarrow 식 또는 값

조건문 : if (조건)

 then (활동)

 else (활동)

반복문 : while (조건)

 do (활동)

알고리즘 단위와 매개변수 : procedure 이름 (매개변수)

7-1 간단한 알고리즘들

유사한 예: 요리 레시피, 유클리드의 호제법, 근의 공식, $n!$ 구하는 알고리즘 등

procedure pH 4로 맞추기

while (pH 수준이 4보다 큼)

do (한 방울의 황산을 추가한다)

procedure Factorial (n)

while (n>1)

do (fac \leftarrow n \times Factorial (n-1))

return 1

7-2. 검색 알고리즘

N개의 수가 순서대로 배열되어 있는 수열에서 특정한 수가 있는지 확인하라.

순차 검색 (최대 N번) vs. 이진 검색 (최대 $\log N$)

7-3. 정렬 알고리즘

N개의 수가 어지럽게 배열되어 있는 수열을 오름차순으로 정렬하라.

버블 소트 (N^2) vs. 퀵 소트 ($N \log N$)

7-4. 좋은 알고리즘 vs. 나쁜 알고리즘

좋은 알고리즘은 실제로 원하는 결과를 정확히 내야 하며, 빠른 시간에 수행해야 함.

더 좋은 알고리즘이 있는지, 원하는 결과를 낼 수 있는 알고리즘이 가능한지는 실제로 만들어 봐야만 알 수 있음.